

**Ollscoil na hÉireann, Gaillimh**  
*National University of Ireland, Galway*  
**Autumn Repeat Examinations 2009**

<b>Exam Code(s)</b>	3BA, 1SD
<b>Exam(s)</b>	3 <sup>rd</sup> Year B.A. (Information Technology) Higher Diploma in Applied Science (Software Design & Development)
<b>Module Code(s)</b>	CT336
<b>Module(s)</b>	GRAPHICS AND IMAGE PROCESSING
Paper No.	
Repeat Paper	
External Examiner(s)	Prof. S. McClean
Internal Examiner(s)	Prof. G. Lyons Dr. S. Redfern
<b><u>Instructions:</u></b>	Answer <b>any three</b> questions. All questions carry equal marks.
<b>Duration</b>	<b>2 hours</b>
<b>No. of Pages</b>	7
<b>Department(s)</b>	Information Technology
<b>Course Co-ordinator(s)</b>	
<b><u>Requirements:</u></b>	
MCQ	
Handout	
Statistical Tables	
Graph Paper	
Log Graph Paper	
Other Material	

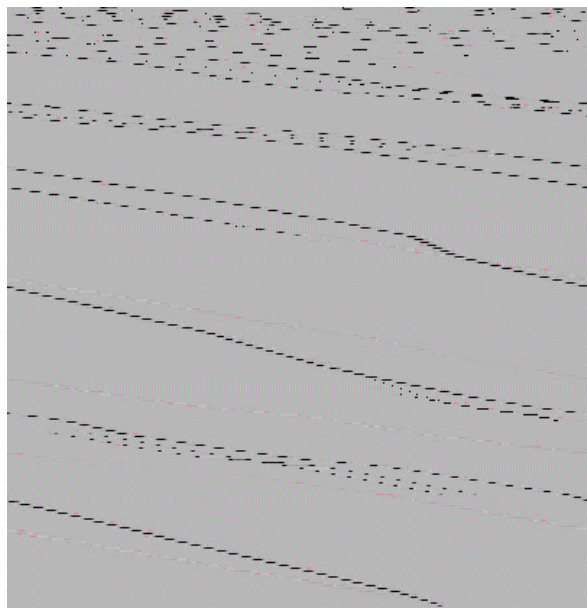
### Q.1.

(a) Many of the techniques used in real-time 3D graphics programming attempt to maximise the realism of the rendered scene while using a minimal number of polygons. With specific reference to this 'polygon budget', and using diagrams where appropriate, discuss each of the following five techniques:

- (i) Back-face culling (2 marks)
- (ii) Texture mapping (2 marks)
- (iii) Bump mapping (2 marks)
- (iv) Skyboxes (2 marks)
- (v) Billboards (2 marks)

(b) The Binary Space Partitioning (BSP) algorithm is widely used in modern graphics programming.

- (i) In what situation is the BSP approach most useful? In what situation is it not useful at all? (2 marks)
- (ii) Consider the diagram below, which depicts a simple 2D scene involving 8 polygons. The polygons are labeled A through H and the arrows indicate their *surface normals*. Construct a BSP tree for this scene, and briefly explain your steps in constructing it. (8 marks)

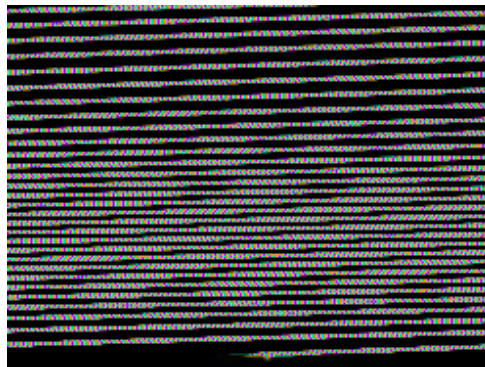


## Q.2.

- (a) With respect to the digital storage of raster (bitmapped) graphics, explain the difference between “lossless” compression and “lossy” compression. Briefly outline the dictionary-based LZW compression algorithm used in GIF image files. What characteristics would you expect to see in an image that is highly suitable for GIF compression? (5 marks).

(b) Extrusion

- (i) Describe the use of extrusion in VRML, referring to each of the seven fields used by the Extrusion node. (5 marks)
- (ii) Write VRML code to produce an extruded vase shape, similar to the one shown below. You should consider its geometry only, and can ignore the use of materials. Note that the most useful VRML nodes, as well as a cross section for this vase, are summarised on the final page of this exam paper. (5 marks)



```
Extrusion
{
  crossSection [ ]
  spine        [ ]
  scale        [ ]
  orientation [ ]
  beginCap
  endCap
  creaseAngle
}
```

- (c) Why are *back buffers* used in real-time graphics applications? Explain their operation. (5 marks).

### Q.3.

(a) In order to set up the projection and camera in a 3D *OpenGL* application, the following two functions are often used:

```
gluPerspective(fov, aspect, near, far);  
gluLookAt(eyex, eyey, eyez, atx, aty, atz, upx, upy, upz);
```

Explain the use of these two functions, in particular the precise meaning of their arguments (6 marks).

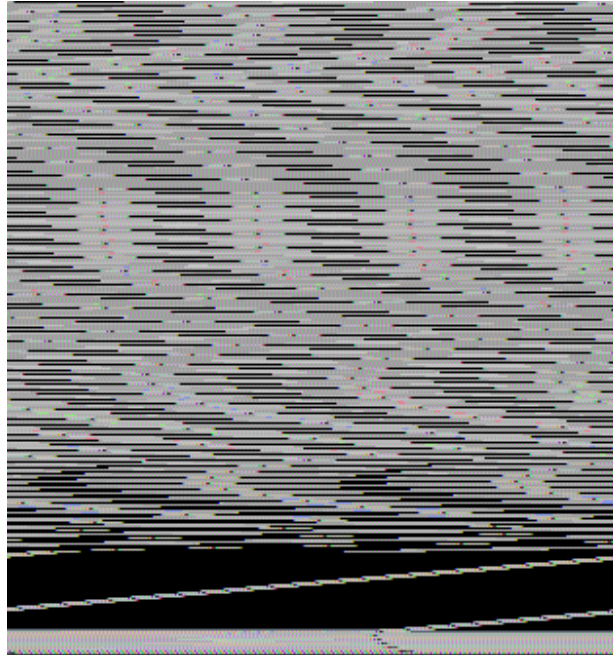
(b) The *OpenGL* program provided below draws a wire cube at the origin. Modify the program so that it uses an *idle callback* function to perform a simple repeating animated rotation of the cube (10 marks).

(c) Further modify the program so that it uses the `gluPerspective` and `gluLookAt` functions to define the projection and camera parameters. Ensure that the values you choose for the arguments to these functions are appropriate so that the cube is visible and centred in the window (4 marks).

```
#include <GL/glut.h>  
void display();  
  
int main(int argc, char** argv)  
{  
    glutInit(&argc,argv);  
    glutCreateWindow("OpenGL Cube");  
    glutDisplayFunc(display);  
    glutMainLoop();  
}  
  
void display()  
{  
    glClear(GL_COLOR_BUFFER_BIT);  
    glMatrixMode(GL_PROJECTION);  
    glLoadIdentity();  
    glFrustum(-1.0, 1.0, -1.0, 1.0, 1.0, 5.0);  
  
    glMatrixMode(GL_MODELVIEW);  
    glLoadIdentity();  
    glutWireCube(1.0);  
    glFlush();  
}
```

**Q.4.**

(a) Write the `display()` function for an *OpenGL* program which renders the following 2D circle using the `GL_TRIANGLE_FAN` primitive. (Note the use of colour to differentiate the triangles in the fan) (8 marks):



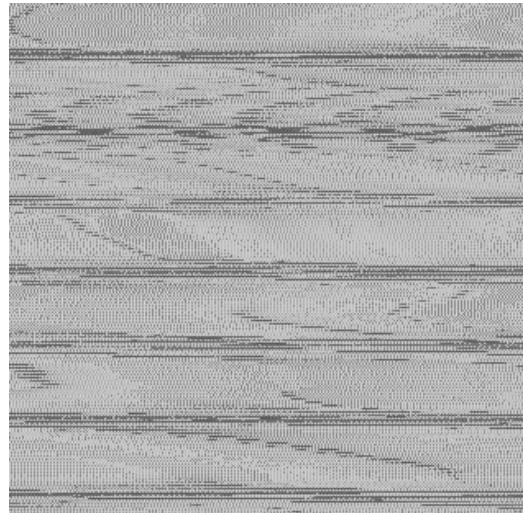
(b) Explain the *keyframe* approach to animation in computer graphics, and explain its use in *VRML*, referring to the *TimeSensor*, *Transform*, *OrientationInterpolator* and *PositionInterpolator* nodes in your answer. (6 marks)

(c) A more powerful approach to producing animations in *VRML* is to use *JavaScript* nodes to dynamically calculate key positions or orientations. Write *VRML* code for a *JavaScript* node which produces a smooth elliptical motion, and which is suitable for use with a *TimeSensor* and a *Transform* node (6 marks).

**Q.5.**

(a) Describe the morphological techniques of erosion and dilation. Compare the four operations (i) opening, (ii) closing, (iii) thinning and (iv) thickening. In what circumstances might each of these four operations be used? (10 marks).

(b) The image on the right is of a printed circuit board (PCB), thousands of which are manufactured every hour in a particular factory. It is required as part of the quality control of this factory to produce an automatic machine vision system, which extracts the traces (straight bits), end points (places at which a trace terminates), and pads (circular bits) in the image. **Present** and **discuss** a suitable and robust set of image processing algorithms for this task (10 marks).



## Some useful VRML node information:

<pre> Shape {     geometry     appearance }  Transform {     children    [ ]     translation 0.0 0.0 0.0     rotation    0.0 0.0 1.0 0.0     scale       1.0 1.0 1.0     center      0.0 0.0 0.0 }  TimeSensor {     enabled          TRUE     startTime        0.0     stopTime         0.0     cycleInterval    1.0     loop             FALSE     isActive         # eventOut     time             # eventOut     cycleTime        # eventOut     fraction_changed # eventOut }  PositionInterpolator {     key    [ ]     keyValue [ ]     set_fraction # eventIn     value_changed # eventOut }  OrientationInterpolator {     key    [ ]     keyValue [ ]     set_fraction # eventIn     value_changed # eventOut }  Extrusion {     crossSection [ ]     spine        [ ]     scale        [ ]     orientation  [ ]     beginCap     endCap     creaseAngle } </pre>	<pre> Box {     size 2.0 2.0 2.0 }  Sphere {     radius 1.0 }  Cylinder {     radius 1.0     height 2.0     side   TRUE     top    TRUE     bottom TRUE }  Appearance {     material }  Material {     diffuseColor     specularColor     ambientIntensity     emissiveColor     transparency     shininess     texture }  ImageTexture {     url }  Co-ordinates for a circle-shaped cross section, suitable for extrusion: 1.00 0.00,    0.92 -0.38, 0.71 -0.71,    0.38 -0.92, 0.00 -1.00,    -0.38 -0.92, -0.71 -0.71,    -0.92 -0.38, -1.00 0.00,    -0.92 0.38, -0.71 0.71,    -0.38 0.92, 0.00 1.00,     0.38 0.92, 0.71 0.71,     0.92 0.38, 1.00 0.00 </pre>
---	---